3
Kaluana

O *middleware* Kaluana original [12] tem como objetivo oferecer ao desenvolvedor de aplicações móveis, maior facilidade na implementação de aplicações dinamicamente adaptáveis. Ele define um modelo de componentes orientado a serviços que permite a composição e implantação de componentes em tempo de execução. Para isto, ele divide o processo de desenvolvimento de aplicações móveis em duas etapas: desenvolvimento de componentes e composição do software. Deste modo, aplicações são formadas pela composição de unidades de implantação reutilizáveis, cujo processo de desenvolvimento é independente do desenvolvimento das aplicações compostas.

Isto permite atrasar a decisão do desenho de uma aplicação fazendo com que o desenvolvedor, ao invés de decidir o comportamento do sistema durante a definição de seu projeto, possa especificar um conjunto de componentes e implementá-los durante o processo o que é chamado de Engenharia de Domínio [52]. Durante a engenharia da aplicação, componentes específicos são desenvolvidos para satisfazer os requisitos, funcionais e não-funcionais, especificados, assim como pode-se fazer reuso dos componentes e da arquitetura de outra aplicação, como parte do software sendo desenvolvido.

Alguns pontos de variação devem ser definidos, para que a aplicação seja adaptativa. Estes são os pontos de conexão, onde os componentes utilizados se conectam com a aplicação. Mais detalhes da arquitetura e do ciclo de vida dos componentes são descritos nas próximas seções.

3.1.
Arquitetura

A arquitetura do *middleware* Kaluana é desenvolvida em cima da plataforma Android [13], a qual provê a ciência de contexto através de provedores que disponibilizam as informações sobre o status do dispositivo (energia, conectividade, etc) e a orientação a serviços. Acima da plataforma Android, se

encontra a camada de *middleware*, a qual possui o gerenciador de adaptações e o gerenciador de componentes, que por sua vez, controla o repositório de componentes do *middleware*, que é onde são registrados todos os componentes que podem ser utilizados pelo sistema.

Segundo [12], o gerenciador de adaptações é responsável por registrar interesse junto a um ou mais provedores de informações de contexto e, a cada mudança significativa no contexto, verificar se existem componentes ou serviços mais adequados ao novo contexto. Para isto, é feita uma busca nos registros do *middleware*, verificando se alguma aplicação possui componentes que não sejam mais adequados com as novas informações do sistema. O gerenciador de componentes por sua vez, é responsável por manter as referências a todos os componentes ativos, bem como atender as requisições de ativação e desativação dos componentes.

O repositório de componentes é responsável por armazenar a referência para todos os componentes disponíveis no dispositivo. Ele é inicializado quando o primeiro componente é registrado no sistema e guarda um mapeamento das políticas de reconfiguração de cada componente para um acesso mais fácil e ágil do comportamento de cada componente. Caso durante uma implantação dinâmica, um componente requisitado por uma aplicação não possua uma entrada no repositório de componentes, o gerenciador de componentes se encarrega de buscar pelo componente em um repositório remoto, que se encontra em um servidor web pré-definido. A Figura 4 mostra a arquitetura do *middleware* Kaluana.

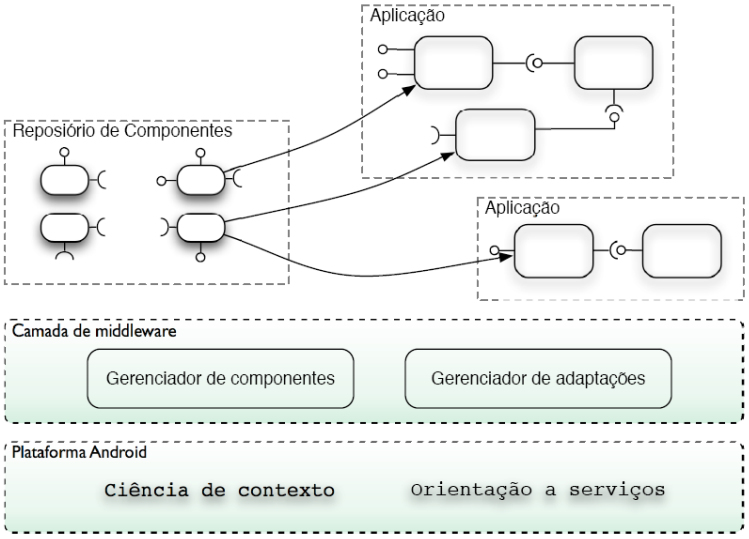

Figura 4. Arquitetura do *middleware* Kaluana, segundo [12]

3.2. Ciclo de vida

De acordo com [12], o ciclo de vida de um componente é composto de cinco etapas bem definidas: Instalação, ativação, utilização, adaptação e desinstalação. O processo de desinstalação, no entanto, não é abordado pelo Kaluana original. A implantação dinâmica é composta pelos processos de instalação, ativação, enquanto a adaptação dinâmica pode implicar nas tarefas de desativação de um componente e ativação de outro, inclusive podendo resultar em uma nova implantação dinâmica, quando o componente não se encontra previamente instalado no dispositivo. A seguir, essas etapas são descritas em mais detalhes:

- **Instalação**

  A *instalação* consiste no desempacotamento e execução do arquivo .APK, específico à plataforma Android, no dispositivo ou no emulador para tornar possível seu uso.

  Um arquivo .APK pode encapsular qualquer quantidade de componentes, além de uma ou nenhuma aplicação. A instalação é realizada no processo de implantação dinâmica, após a realização do *download* do arquivo que contém o componente buscado em um repositório remoto, caso seja necessário. O processo de instalação foi alterado nesse trabalho no sentido de que agora a instalação é realizada em dois passos: registro e carregamento de componentes. O primeiro só registra o contrato de reconfiguração de um componente no repositório local do Kaluana enquanto o segundo carrega componente na memória, fazendo sua instanciação para que a aplicação possa usá-lo. Este processo é descrito em detalhes na seção 0 e 5.6.

- **Ativação**

  A ativação consiste na instanciação da classe que representa o componente e na criação, em tempo de execução, dos seus serviços e receptáculos de acordo com as definições das interfaces públicas dos componentes, feitas em tempo de desenvolvimento.

  O processo de ativação pode ocorrer como resultado da solicitação de um componente pela aplicação ou de uma adaptação dinâmica. No segundo caso, o processo deve implicar na criação das mesmas

conexões que o componente anterior mantinha, bem como no restabelecimento do estado interno similar ao do componente substituído.

- **Utilização**

O desenvolvedor de uma aplicação pode verificar os serviços e receptáculos disponíveis e realizar conexões entre componentes. Para isso, o desenvolvedor pode utilizar métodos providos pelo próprio componente. Estes métodos são implementados pelo *middleware* Kaluana de forma transparente para o desenvolvedor do componente.

Uma vez escolhido um serviço e um receptáculo, é necessário mais um passo para conectá-los. O desenvolvedor deve usar o método *getService(String serviceName)* para recuperar uma referência para o *stub* do cliente do serviço no processo executado pelo requisitante e passá-lo ao componente que contém o receptáculos por seu método *bindReceptacle()*.

Uma vez realizadas todas as conexões, componentes passam a manter referências a serviços de outros componentes, assim podem interagir para realizar as tarefas a que a aplicação se propõe.

A partir do momento em que a aplicação realiza a conexão entre um receptáculo e um serviço, o *middleware* associa, por meio de reflexão computacional, a implementação do serviço no componente provedor ao receptáculo na classe cliente. Ou seja, o desenvolvedor do componente não precisa se preocupar com a descoberta dos serviços, uma vez que as implementações serão injetadas pelo Kaluana, em tempo de execução, através da instanciação e conexão dos serviços e receptáculos do componente. Basta indicar quais são os receptáculos do componente e o *middleware* se encarrega de instanciar os serviços mais adequados (que possuem menor quantidade de restrições computacionais para sua execução), segundo as conexões realizadas pela aplicação.

Nenhum código de inicialização dos serviços ou receptáculos é implementado pelo desenvolvedor do componente. Toda a lógica de inicialização é controlada pelo *middleware* Kaluana. A única exigência é a implementação de um método *setter*, que recebe uma referência para

a interface que implementa os receptáculos necessários para a execução do componente. Esta interface será instanciada pelo Kaluana com a implementação do serviço necessário, de acordo com os componentes registrados no repositório, através do critério de escolha definido no *middleware*. Este critério é feito de acordo com a comparação das restrições de execução de cada componente e é descrito de maneira aprofundada na seção 5.1.

- **Desativação**

Neste processo, todos os serviços do componente são paralisados e seus receptáculos desconectados de serviços remotos. Em seguida, todos os seus serviços são finalizados explicitamente pelo *middleware* Kaluana. O componente, entretanto, continua instalado no dispositivo móvel, para o caso de ser usado posteriormente.